Offload Decision Models and the Price of Anarchy in Mobile Cloud Application Ecosystems

Bo Gao, Ligang He and Stephen A. Jarvis
Department of Computer Science, University of Warwick, Coventry, CV4 7AL, UK,
{bogao, liganghe†, saj}@dcs.warwick.ac.uk

***Abstract*—With the maturity of technologies such as HTML5 and JavaScript, and with the increasing popularity of cross-platform frameworks such as Apache Cordova, mobile cloud computing as a new design paradigm of mobile application developments is becoming increasingly more accessible to developers. Following this trend, future on-device mobile application ecosystems will not only comprise a mixture of native and remote applications, but also include multiple hybrid mobile cloud applications. The resource competition in such ecosystems and its impact over the performance of mobile cloud applications has not yet been studied. In this paper, we study this competition from a game theoretical perspective and examine how it affects the behaviour of mobile cloud applications. Three offload decision models of cooperative and non-cooperative nature are constructed and their efficiency compared. We present an extension to the classic load balancing game to model the offload behaviours within a non-cooperative environment. Mixed-strategy Nash equilibria are derived for the non-cooperative offload game with complete information which further quantifies the price of anarchy in such ecosystems. We present simulation results which demonstrate the differences between each decision model's efficiency. Our modelling approach facilitates further research in the design of the offload decision engines of mobile cloud applications. Our extension to the classic load balancing game broadens its applicability to real-life applications.**

***Index Terms*—Mobile computing, mobile cloud computing, energy-aware**

I. Introduction

Mobile cloud computing is an emerging field of research that aims to provide a platform on which intelligent and feature-rich applications are delivered to the user's fingertips efficiently. This efficiency comes from the adaptive offload ability of mobile cloud applications which is key to the seamless integration of mobile devices and cloud servers. Pioneered by the likes of MAUI [1], CloneCloud [2] and ThinkAir [3], adaptive computation offload as a core technology in mobile cloud computing has gathered momentum in recent years and has grown from a futuristic concept to a practical means to improve and augment the user's experience of mobile applications.

A mobile cloud application as we discuss in this paper is an application whose main functionality may be executed independently on either a mobile device or a cloud server. This means that the application is able to offload or migrate itself seamlessly between the two platforms. This offload decision is often taken at runtime according to the current network condition and the anticipated workload size [1], [4]. We also refer to this class of applications as *H*ybrids as opposed to *N*ative and *R*emote to distinguish applications by their designated execution platform.

It is important to note that our use of these three terms, hybrid, native and remote, refers to the place of execution of the application's main functionality and its ability to seamlessly migrate between device and cloud, rather than the traditional use of these terms where they refer to the environment it is developed in. Traditionally when an application is written in a native language like Objective-C for iOS devices or Java for Android devices, it is referred to as a native application; an application that's run on a web server and delivered to the user via a browser is referred to as a remote mobile web application. A hybrid application in this sense is a crossover between these two approaches. The majority of a hybrid application's code is usually written in HTML5 and JavaScript and rendered by the device's web engine, so the code is portable between platforms. A hybrid application also include native codes to refine user experience and get access to a wider range of device functionalities. This code portability is an attractive option for the development of mobile cloud applications. However, a hybrid application as in mobile cloud computing is more intelligent in utilising different platforms at runtime.

In order to qualify as a hybrid application as we discuss in this paper, the application need not only be deployable to different platforms, but also make offload decisions at runtime to improve user experience. The code portability of a hybrid mobile cloud application also need not be limited to the use of HTML5, MAUI [1] is written in C# for Microsoft's .NET Common Language Runtime, CloneCloud [2] modified the Dalvik VM for code migration on Android OS, ThinkAir [3] builds its offload platform with a modified version of Android x86. A more recent work [5] utilises a modified version of WebKit to support the offload of HTML5 workers.

Besides existing research level implementations of mobile cloud applications (we recommend two excellent surveys, [6] and [7], to the interested readers for a comprehensive list of existing research in mobile cloud computing), we argue that the increasing popularity of HTML5 as a mobile application development framework also greatly shortens the time required to develop applications that are deployable both natively on the mobile device and remotely as cloud services. The use of HTML5 and platforms like Apache Cordova help significantly lower the level of technical challenges involved in the development of mobile cloud applications. We expect

† Dr. Ligang He is the corresponding author.

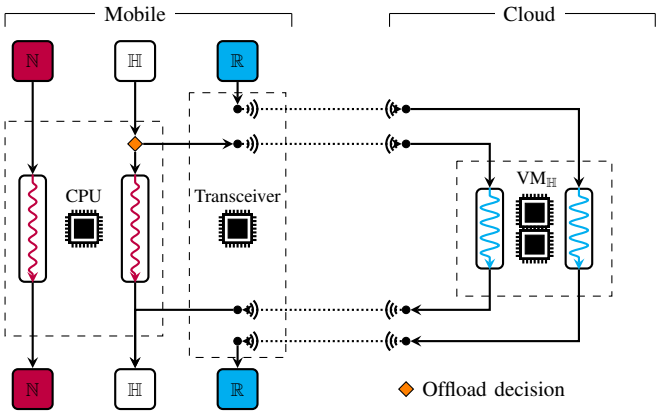

Fig. 1. A mobile cloud application ecosystem

to see an increasing number of mobile applications to adopt the adaptive execution approach proposed by the research of mobile cloud computing in the near future.

A. Problem Statement

With the increasing popularity of mobile cloud applications come one problem currently missing from the research of mobile cloud computing which is the recognition of the competition for resources between applications on mobile devices. Applications are selfish entities. Notwithstanding the cooperative interactions that may exist within certain application workflows, given a host device, each application's performance is proportional to the exclusivity it has over its host's resources. Therefore the competition for resources underlies each community of applications that lives on the same computing device. Recognising the existence of this competition is especially important for the applications that are hosted by resource constrained mobile devices.[1]

We illustrate the resource competition in a mobile cloud application ecosystem with Fig. 1. Three classes of applications share the same mobile device. A wireless connection is established to a remote cloud service supporting computation offload[2]. The main functionality of a native application is carried out on the local CPU, whereas a remote application carry out the majority of its computation via cloud services. To access a cloud service, data is sent via the transceiver of equipped on the mobile device. A hybrid application has the ability to choose between the two platforms. Its offload decision precedes the execution of its main functionality.

Competition of resources comes with either options for a hybrid application. The path of native execution is shared with other native applications at the CPU, whereas the path of remote execution is firstly bottlenecked at the transceiver, and consequently congested at the supporting cloud server. This competition is apparent between hybrids and other two classes of applications, but more importantly it exists within the hybrid class itself.

Existing research in mobile cloud computing focuses on the application's ability to offload computation between mobile and cloud. Offload decisions in existing work are based on the device's parameters without taking into account that it may not be the only application that's using these resources (i.e. the processing unit and the wireless data connection). This uninformed decision making process means that the offload decision made may not be as beneficial as predicated.

TABLE I
EFFECT OF DIFFERENT OFFLOAD DECISIONS.

	Scenario A		Scenario B		Scenario C	
$i \in [s]^{\mathbb{H}}$	$\mathbb{N}$	$\mathbb{R}$	$\mathbb{N}$	$\mathbb{R}$	$\mathbb{N}$	$\mathbb{R}$
$i = 1$	15	(10)	(15)	10	15	(10)
$i = 2$	18	(15)	18	(15)	(18)	15
Cost/Platform	0	(25)	15	15	(18)	15
Social Cost	25		15		18	

B. A Simple Example

We demonstrate the effect of an uninformed offload decision with a simple example as shown in Table I. We assume three scenarios where two hybrid applications share a device. Each number in the table represents the amount of time it takes the application to run on a platform assuming exclusive usage of the device's resources, in seconds. A circle represents the decision made by the application. Scenario A is a typical example of applications making uninformed decisions. Both applications assume that it is the only application running on the device and the cost comparison between the two platforms means both applications prefer to execute on the cloud. This makes $\mathbb{R}$ congested while leaving the $\mathbb{N}$ vacant. The total cost on $\mathbb{R}$ is 25 seconds compared to the cost of 0 on $\mathbb{N}$.

From the user's point of view, the makespan (i.e. social cost as we discuss in detail in III-C4 and III-D) of the system as a whole is 25 seconds. This social cost is higher than either of the other scenarios where the applications' choice of platform are split between $\mathbb{N}$ and $\mathbb{R}$.

From each application's point of view, in A, if each application's sub tasks are scheduled in a round-robin way on $\mathbb{R}$, the expected time costs for both applications are 25 seconds; if the scheduling order is randomly chosen between the two applications as a whole, the expected cost is 17.5 seconds for

[1]Note that the application of our approach is not limited to communities of hybrid applications in future developments. With the popularity of mobile applications (or apps in short), the real estate of a mobile device has already been heavily competed on by the many apps that are currently installed on each device. According to the data published by Google's Our Mobile Planet report [8] for 2013, on average 28.5 apps are installed on each smartphone in the UK which is just above the overall average (26) among the 47 countries included in the survey. In South Korea and Switzerland this number is higher at 40 apps per smartphone. A similar figure is reported by Nielsen in their early 2014 report [9] which includes both Android and iOS users. [10] report an average number of 177 apps installed on their participant's android devices.

[2]A remote application does not have to run on the same cloud server as the hybrid applications. A proprietary application (e.g. Facebook or Twitter) is usually supported by its own servers. Furthermore, a proprietary server is also unlikely to accept offload requests from a personal device. For these type of remote applications, we set w_i^b to be zero in our model since they don't consume the computation resources on our cloud.

i = 1, and 20 seconds for i = 2, all higher than the cost if it were run on $\mathbb{N}$.

C. Objective and Contribution

In this paper, we model each of the three offload decision models that applies to a mobile cloud computing scenario in Sec. III. We especially focus on the game theoretical modelling of the offload game with complete information. We derive the mixed-strategy Nash equilibrium of the game and its social cost at equilibrium in Sec. III-C. The derivation of the Nash equilibrium is significant that it provides a basis for measuring the distance (referred to as the "price of anarchy" of the game which we introduce in Sec. III-C4) between a non-cooperative and a cooperative application ecosystem. With the model we present in Sec. III-C, we also extend the classic load balancing game [11] which has been highly cited since its publication. Comprehensive simulation experiments has been conducted and presented in Sec. IV. Results from the comparisons between the three models provide us with a rare insight into the behaviours of applications within a community (ecosystem).

The impact and future direction of this paper is in two folds. First, from the user's perspective, we provide a suite of modelling tools to quantify the costs and benefits of different offload decision making processes so that an informed decision can be made on a global level. Our results pave the way for future development of manager services of hybrid applications on the device to provide a cooperative environment. Second, from a hybrid application's point of view, in absence of a cooperative mechanism, it is able to derive an offload strategy that's most beneficial to itself.

II. Related Work

Our work furthers existing research of mobile cloud computing by modelling the resource competitions in such application ecosystems and analysing its effects over the efficiency of the offload actions of mobile cloud application. We study this competition from a game theoretical perspective. In this section, we discuss related work in the area of mobile cloud computing.

The idea of transferring computation to a nearby processing unit in order to improve mobile application's performance and reduce local energy cost has been researched along with the maturity of mobile technologies. Many ideas and techniques we use in this paper are inspired by this work.

Early research focuses on the partitioning schemes of an application. Aimed at energy management, a compile-time framework supporting remote task execution was first introduced in [12]. Based on the same approach, a more detailed cost graph was used in [13] with a parametric analysis on its effect at runtime presented in [14]. Another compiler-assisted approach was introduced in [15], which turns the focus to reducing the application's overall execution time. Spectra [16] adds application fidelity (a run-time QoS measurement) into the decision making process and uses it to leverage execution time and energy usage in its utility function. Spectra monitors the hardware environment at run-time and choose between programmer pre-defined execution plans. Chroma [17] builds on Spectra but constructs the utility function externally in a more automated fashion. MAUI [1] also reduces the programmer's workload by automating some of the partitioning process models. The offload decision engine applies an integer programming techniques to produce allocation schemes. Aimed at reducing the communication costs, [18] proposes the concept of cloudlets, which brings the distant Cloud to the more commonly accessible WiFi hotspots. A dynamic VM synthesis approach is proposed in [18]. The offload decision models (section III-B) in these studies estimate the benefit of an offload action based on the device's current bandwidth to the network.

Besides existing research level implementations of mobile cloud applications (we recommend three excellent surveys, [6], [7] and [19], to the interested readers for a comprehensive list of existing research in mobile cloud computing), we argue that the increasing popularity of HTML5 as a mobile application development framework also greatly shortens the time required to develop applications that are deployable both natively on the mobile device and remotely as cloud services. The use of HTML5 and platforms like Apache Cordova help significantly lower the level of technical challenges involved in the development of mobile cloud applications. We expect to see an increasing number of mobile applications to adopt the adaptive execution approach proposed by the research of mobile cloud computing in the near future.

Our work is distinguishable from existing studies in that we considers the efficiency of the application ecosystem on each device rather than that of a single application. The cooperative offload decision model we propose take into account the resource competition between applications within each ecosystem which is missing from existing research. The ability to offload or migrate computation from mobile devices to clouds is integral to the research of mobile cloud computing. Before an offload decision is made, the application must estimate the potential cost and benefit of such an action. When an application is assumed to have exclusivity or strict top priority over its host device's wireless data connection, the application may estimate its offload cost based on the device's entire bandwidth rather than the actual share of bandwidth available on the host device. This causes the application to over-estimate the benefit of an offload action.

III. Decision Models of Computation Offload

In a mobile cloud computing scenario, applications have the option to either execute locally on its host device or offload and execute remotely on a supporting cloud platform. The application must estimate the cost and benefit of an offload action prior to making a decision. Depending on how much information this application has of other applications running on the same device, this decision making process may yield different results. In this section, we formulate the different decision models of a mobile cloud application ecosystem.

We begin with an introduction of the notations used to describe a mobile cloud application ecosystem.

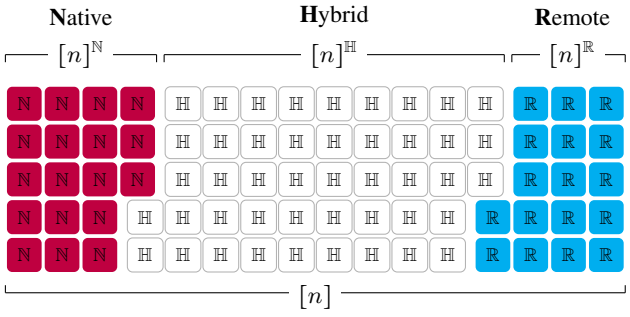

Fig. 2. Composition of the Mobile Application Ecosystem

A. System Notations of Mobile Cloud Application Offload

To describe a mobile cloud application ecosystem, we assume a set of n independent applications sharing the same mobile device, denoted as $[n] = \{1, \ldots, n\}$. Each application $i \in [n]$ is to choose between two parallel execution platforms, which we refer to as the remote cloud $\mathbb{R}$ and the native processing unit $\mathbb{N}$ in our model, in order to minimise its execution time cost.

Let a_i^j be a binary variable indicating i's decision to execute on platform j. All a_i^j together constitute an assignment $A : [n] \to \{\mathbb{R}, \mathbb{N}\}$ with $A(i)$ denotes the chosen platform for i.

The weight of each application i has two components:

- w_i^d which denotes the size of the **d**ata that is to be transmitted over the wireless network if application i is offloaded to $\mathbb{R}$,
- w_i^b which denotes the amount of computation **b**inary that is associated with i.

In correspondence, the speed in which each platform $j \in \{\mathbb{R}, \mathbb{N}\}$ can process an application also consists of two components:

- s_j^d which denotes the data transmission speed[3] to j, with $s_{\mathbb{N}}^d = inf$ and $s_{\mathbb{R}}^d$ = bandwidth between $\mathbb{N}$ and $\mathbb{R}$,
- s_j^b which denotes the computation speed of j's processing unit, we assume $s_{\mathbb{N}}^b < s_{\mathbb{R}}^b$.

Not all mobile applications in $[n]$ has the ability to migrate between $\mathbb{N}$ and $\mathbb{R}$. Some are fixed to run natively (locally), whereas some may rely on an active data connection to run remotely. To represent this distinction within $[n]$, we divide $[n]$ into three distinct subsets:

- $[n]^{\mathbb{N}}$ for native applications fixed to run on $\mathbb{N}$,
- $[n]^{\mathbb{R}}$ for remote applications fixed to run on $\mathbb{R}$,
- $[n]^{\mathbb{H}}$ for hybrid (mobile cloud) applications that may run on either $\mathbb{N}$ or $\mathbb{R}$.

This composition of applications is illustrated by Fig. 2.

Note that we use subscripts for applications and superscripts for platforms when a variable is associated with both sets. With these notations, we first derive the classic offload decision model.

[3]When an application is run on the local device, we assume that the speed at which its binary reaches the processor is infinite. This way we keep the equations generic, and we don't have to add an indicator variable inside the subsequent equations (e.g. (2)).

B. Offload with Symmetrically Incomplete Information

In this scenario, each application i knows the properties of both platforms ($s_j^d,\ s_j^d,\ j \in \mathbb{N}, \mathbb{R}$) and of its own task ($w_i^d,\ w_i^b$), but is unaware of the other applications who also share the resources provided by the same device. Due to this limitation, exclusive usage of the device's data connection and processor is assumed by all applications. Hence the offload decision of i is given by

$$a_i^{\mathbb{R}} = \begin{cases} 1, & \text{If } \dfrac{w_i^d}{s_{\mathbb{R}}^d} + \dfrac{w_i^b}{s_{\mathbb{R}}^b} < \dfrac{w_i^d}{s_{\mathbb{N}}^d} + \dfrac{w_i^b}{s_{\mathbb{N}}^b} \\ 0, & \text{Otherwise.} \end{cases} \tag{1}$$

with $a_i^{\mathbb{N}} = 1 - a_i^{\mathbb{R}}$.

Depending on the capacity of the device's wireless data connection ($s_{\mathbb{R}}^d$), the benefit of remote execution (i.e. reduced execution time, given by $w_i^b/s_{\mathbb{N}}^b - w_i^b/s_{\mathbb{R}}^b$) may be offset by the additional communication cost (between the device and the cloud, given by $w_i^d/s_{\mathbb{R}}^d$) when applications are run on or offloaded to the cloud. Therefore mobile cloud applications often require that the data connection speed between $\mathbb{N}$ and $\mathbb{R}$ to be greater than a certain threshold before an offload action is considered [4], [20]. The capacity of the device's wireless data connection greatly influence the decision making process of offload-able applications.

There are two potential flaws in this offload decision model. First, the wireless connection may be occupied by other applications, which means that the actual data transmission cost is greater than $w_i^d/s_{\mathbb{R}}^d$. Second, the local processing unit is also shared with other applications, hence the cost of local execution $w_i^b/s_{\mathbb{N}}^b$ and therefore the benefit of remote execution as given by $w_i^b/s_{\mathbb{N}}^b - w_i^b/s_{\mathbb{R}}^b$ are also under-estimated. Both flaws are direct results of the incomplete information given to each application.

To complete the notation of this subsection, we denote Θ_B to represent the social cost of the system under the symmetrically incomplete information decision model. We further discuss the definition of social costs in III-C4 and III-D. The "B" in this notation comes from the fact that the wireless data bandwidth plays a crucial role in this decision model. Next, we derive the decision model when applications are given complete information of other applications.

C. Offload with Complete Information

We now consider the scenario in which all applications are given complete information of the weights[4] of all other applications ($w_i^b,\ w_i^d,\ i \in [n]$). Given an assignment $A : [n] \to \{\mathbb{R}, \mathbb{N}\}$, the cost (time delay) for application i is given by

$$c_i = \sum_{\substack{k \in [n] \\ A(k) = A(i)}} \left(\frac{w_k^d}{s_{A(k)}^d} + \frac{w_k^b}{s_{A(k)}^b} \right) \tag{2}$$

which assumes that no priority is assigned to any application. That is to say that both data packets over the data connection and instructions in the processor stack are scheduled in a

[4]In practical terms, the weights of an application can be predicted based on its historic profiles as done in [10].

round-robin way. Following this, the cost of platform j is given by

$$C_j = \sum_{\substack{i\in[n]\\A(i)=j}} \left(\frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} \right) \tag{3}$$

(2) and (3) together correct the inaccuracy caused by incomplete information.

1) The Offload Game: It is easy to see that the decision of each application is directly influenced by the decisions made by others. Since each application's goal is to minimise its own cost, the offload decision model with complete information can be described by a non-cooperative game theoretic framework.

In this game, which we refer to as the *offload game*, each application is an agent (player) whose objective is to minimise c_i. Each application has a strategy profile of $\{\mathbb{N}, \mathbb{R}\}$. A collection of pure strategies of all applications $i \in [n]$ constitutes an assignment A. A *mixed strategy* [5] is a probability distribution over the set of pure strategies $\{\mathbb{N}, \mathbb{R}\}$.

2) Mixed Strategies and Expected Costs: We first denote the probability that agent i choose to run on platform j with $p_i^j = \mathbb{P}[A(i) = j]$. Then the expected cost of platform j under the *strategy profile* $P = \{p_i^j,\ i \in [n],\ j \in \{\mathbb{N}, \mathbb{R}\}\}$ is

$$\mathbb{E}[C_j] = \sum_{i\in[n]} p_i^j \left(\frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} \right). \tag{4}$$

For application i, its expected cost when selecting j is

$$\mathbb{E}[c_i^j] = \frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} + \sum_{\substack{k\in[n]\\k\neq i}} p_k^j \left(\frac{w_k^d}{s_j^d} + \frac{w_k^b}{s_j^b} \right). \tag{5}$$

This together with (4), we have

$$\mathbb{E}[c_i^j] = \mathbb{E}[C_j] + (1 - p_i^j) \left(\frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} \right) \tag{6}$$

which derives

$$p_i^j \left(\frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} \right) = \mathbb{E}[C_j] - \mathbb{E}[c_i^j] + \left(\frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} \right) \tag{7}$$

and further derives

$$p_i^j = \left(\mathbb{E}[C_j] - \mathbb{E}[c_i^j] + \left(\frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} \right) \right) \Big/ \left(\frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} \right) \tag{8}$$

which gives all applications' mixed strategies as a function of $\mathbb{E}[C_j]$ and $\mathbb{E}[c_i^j]$ and constitutes P.

[5]We consider mixed strategies rather than pure strategies because it is a better match to the mobile cloud computing scenario. First, in a game, there may be multiple (or none as in the rock-paper-scissors game) pure strategy equilibria, including the optimal assignment which we derive in subsection III-D. To reach a pure strategy equilibrium, the order in which each agent is given the right to make a strategy decision affects which pure strategy equilibrium the system would reach. In our mobile cloud scenario, the mobile OS does not explicitly define this order, and it also wouldn't be fair for the OS to do so without user consent. Second, beside the saving in execution time, hybrid applications can also provide the user with higher quality service when it is run on a remote cloud as seen in [21]. Therefore, the user may opt for a remote execution regardless. Therefore, only a probability of an application's pure strategy can be observed. Because of these reasons a pure strategy profile is not a stable representation of our offload game.
On the contrary, a mixed strategy profile only requires that each application is aware of the probability of others' offload decisions. The mobile OS has this information readily available from its network access log, and is able to share this with all applications.

3) Nash Equilibrium: We now describe the Nash equilibrium of this game. A game is said to be in Nash equilibrium when no agent (application i) of the game, with complete knowledge of all other agents' strategies (P), is able to make gains or reduce its cost by unilateral actions. Not all strategy profiles define a Nash equilibrium. In order to find the P which defines a Nash equilibrium, further constraints is to be added to (8).

First, in a Nash equilibrium, each application agent only assign non-zero probabilities to platform j if

$$\mathbb{E}[c_i] = \mathbb{E}[c_i^j] = \min_{j\in\{\mathbb{N},\mathbb{R}\}} \mathbb{E}[c_i^j],\ i \in [n]. \tag{9}$$

We define a support indicator

$$\alpha_i^j = \begin{cases} 1, & \text{if } p_i^j > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

Take (7) into (4) with the introduction of α_i^j and (9), we get

$$\mathbb{E}[C_j] = \sum_{i\in[n]} \alpha_i^j \left(\mathbb{E}[C_j] - \mathbb{E}[c_i] + \left(\frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} \right) \right) \tag{11}$$

for $j \in \{\mathbb{N}, \mathbb{R}\}$.

Second, each application i should distribute all of its weight completely, that is

$$\sum_{j\in\{\mathbb{N},\mathbb{R}\}} p_i^j = 1, \quad i \in [n]. \tag{12}$$

Take (8) into (12) with the introduction of α_i^j and we get

$$\sum_{j\in\{\mathbb{N},\mathbb{R}\}} \alpha_i^j \left(\mathbb{E}[C_j] - \mathbb{E}[c_i] + \left(\frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} \right) \right) = \left(\frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b} \right) \tag{13}$$

for $i \in [n]$.

Observe that (11) and (13) together have $n + 2$ variables ($\mathbb{E}[C_j]$ and $\mathbb{E}[c_i]$) and $n+2$ equations, meaning that a unique solution is defined. Therefore, the strategy profile of the Nash equilibrium of our offload game is completely defined by (8), (11) and (13). We further give the solution of $p_i^{\mathbb{R}}$ in (14) with $C_{\mathbb{R}}^f$ and $C_{\mathbb{N}}^f$ denote the cost from $[n]^{\mathbb{R}}$ and $[n]^{\mathbb{N}}$ respectively. The corresponding derivation is attached in Appendix A.

4) Social Cost, Expected Makespan at P and Price of Anarchy: So far we have been looking at the costs from each application's perspective. Indeed, because of the non-cooperative nature of the offload game, the derivation of P is driven by each application's expected c_i. However, from user's perspective, the overall cost of the system is of greater importance. In game theory terms, this system cost is referred to as the *social cost* of the game system. In our offload game, we define the social cost to be the makespan of the system. We discuss the optimal social cost in the next subsection (III-D) with the cooperative decision model. But first, following our results of the Nash equilibrium strategy profile P, we derive the social cost of the system at Nash equilibrium.

Given a strategy profile P we derive the social cost (expected makespan) of the system at P, which we denote with Θ_P as

$$\Theta_P = \sum_{A(1)\in\{\mathbb{N},\mathbb{R}\}} \cdots \sum_{A(n)\in\{\mathbb{N},\mathbb{R}\}} \prod_{i=1}^{n} p_i^{A(i)} \max_{j\in\{\mathbb{N},\mathbb{R}\}} \mathbb{E}[C_j] \tag{15}$$

$$p_i^{\mathbb{R}} = \left(\frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b}\right) \Bigg/ \left(\frac{w_i^d}{s_{\mathbb{N}}^d} + \frac{w_i^b}{s_{\mathbb{N}}^b} + \frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b}\right) + \left(C_{\mathbb{R}}^f - C_{\mathbb{N}}^f + \sum_{k \in [n]^{\mathbb{H}}} \left(\frac{w_k^d}{s_{\mathbb{R}}^d} + \frac{w_k^b}{s_{\mathbb{R}}^b}\right) - \left(\frac{w_k^d}{s_{\mathbb{N}}^d} + \frac{w_k^b}{s_{\mathbb{N}}^b}\right)\right) \Bigg/ \left(\left(1 - |[n]^{\mathbb{H}}|\right)\left(\frac{w_i^d}{s_{\mathbb{N}}^d} + \frac{w_i^b}{s_{\mathbb{N}}^b} + \frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b}\right)\right) \tag{14}$$

This quantity gives an indication of the system's performance at P. When strategy profile P defines an equilibrium, it is important to compare Θ_P (*Nash social cost*) with the system's optimal performance (optimal social cost), denoted as Θ_{opt} which we discuss in the next subsection (III-D). The ratio $\Theta_P : \Theta_{opt}$ is referred to as the *price of anarchy* (also referred to as "coordination ratio" in [11]) of the game.

We study the price of anarchy of a system which is an indication of how much worse a system would perform if no control is applied on a system level. First introduced in [11], price of anarchy is a key concept often associated with the study of Nash equilibrium in game theory. A Nash equilibrium as we have shown is driven by the selfish behaviours of the agents of a system. Because each agent is only concerned with its own cost when making strategy decisions, without system level control, the overall performance of the system in anarchy becomes a by-product of the competition between the agents. The distance between this by-product and the optimal performance is represented by the price of anarchy of the game.

We show in the following subsection that the system cost can be minimised when system level control is applied. Then in IV-B and IV-C we further demonstrate how system performance is described by price of anarchy.

D. Cooperative Offload and the Makespan Scheduling Problem

The offload decision models we discussed in the previous two subsections both assume non-cooperative behaviours within the system. In this third offload decision model, we assume the contrary where a global authority is in place to manage the offload / migration behaviour of the mobile cloud application ecosystem.

From a global perspective, recall that the cost of the system (also referred to as *social cost* in game theory terms) is defined to be the makespan, that is, the maximum schedule length between the two platforms. This naturally leads to a variation of the classic makespan scheduling problem. Recall that a_i^j indicates if i chooses to run on j, and that $[n]^{\mathbb{N}}$ and $[n]^{\mathbb{R}}$ denote the subsets of applications that are fixed to run on $\mathbb{N}$ and $\mathbb{R}$ respectively. With these we formulate the problem as an integer program:

$$\text{minimise} \quad \Theta_{opt} = \max_{j \in \{\mathbb{R},\mathbb{N}\}} \sum_{i=1}^{n} a_i^j \left(\frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b}\right) \tag{16}$$

$$\text{subject to} \quad a_i^{\mathbb{R}} + a_i^{\mathbb{N}} = 1, \quad i \in [n] \tag{17}$$

$$a_i^j \in \{0,1\}, \quad i \in [n],\ j \in \{\mathbb{R},\mathbb{N}\} \tag{18}$$

$$a_i^{\mathbb{N}} = 1, \quad i \in [n]^{\mathbb{N}} \tag{19}$$

$$a_i^{\mathbb{R}} = 1, \quad i \in [n]^{\mathbb{R}}. \tag{20}$$

Note that our problem is different from the classic makespan scheduling problem in that the speed of each machine (platform) consists of two sub-speeds ($s_j = \{s_j^d, s_j^b\}$). Therefore the machines in our problem can not be ordered by their speeds as in the classic makespan scheduling problem [22]. The complexity of this problem is at least NP-hard since it contains a special case, when $\forall j \in \{\mathbb{R},\mathbb{N}\} : s_j^d = s_j^b$, which can be reduced to a classic makespan scheduling problem which is NP-hard even for two identical machines.

The solution of this integer program gives us the optimal assignment in terms of minimising the social cost of the system. However, besides the complexity, the solution also assumes that there is a global authority that enforces the assignment which is not the case in the current mobile cloud computing framework. Operating systems who manage the wireless data protocol on mobile devices does not schedule where applications are run. Techniques exist to exploit delay-tolerant property of some applications to reduce the tail energy overhead [23]. Pre-fetching is another technique used to improve the efficiency of the data link [24], [23]. Though in all cases, the operating system attempts to complete all requests from applications and does not proactively seek to offload any particular application.

Existing offload techniques in mobile cloud computing assumes exclusivity over the host device's data link. Offload decisions are made selfishly by the application. Therefore we next introduce a game theoretic framework to study the effect of the selfish behaviours in the ecosystem of mobile cloud applications.

IV. Experiments

In this section we demonstrate and visualise the behaviours of mobile cloud applications under different offload decision models, and the influence of such over the social cost of mobile cloud application ecosystems.

Each group of simulation tests is referred to in this paper by a group ID which is given in the first column of Table II. Detailed parameters of these test groups are also given in this table. We define each application's data and

TABLE II
SIMULATION PARAMETERS

Test		Application Parameters					Platform Parameters
Group	Cycles	$[\lvert[n]^{\mathbb{N}}\rvert, \lvert[n]^{\mathbb{R}}\rvert, \lvert[n]^{\mathbb{H}}\rvert]$	Support	$[w_i^d, w_i^b]$	Observed	$[w_k^d, w_k^b]$	$[s_{\mathbb{N}}^d, s_{\mathbb{N}}^b, s_{\mathbb{R}}^d, s_{\mathbb{R}}^b]$
S1	40	$[0,0,10]$	$i \in \{2,\ldots,10\}$	$[50,500]$	$k \in \{1\}$	$[50,(+10)500^{\dagger}]$	$[inf,200,50,800]$
S1F	40	$[0,0,10]$	$i \in \{2,\ldots,10\}$	$[50,500]$	$k \in \{1\}$	$[50,(+10)500]$	$[inf,200,50,1600]$
S2	40	$[0,0,10]$	$i \in \{2,\ldots,10\}$	$[50,500]$	$k \in \{1\}$	$[50,500(+10)]$	$[inf,200,20,800]$
S3	40	$[0,0,10]$	$i \in \{2,\ldots,10\}$	$[50,(+10)500]$	$k \in \{1\}$	$[50,500]$	$[inf,200,50,800]$
S4	40	$[0,0,10]$	$i \in \{2,\ldots,10\}$	$[50,500(+10)]$	$k \in \{1\}$	$[50,500]$	$[inf,200,20,800]$
Y1	200	$[1,1,15]$	$i \in [n]$	$[50,Expo(500)]$	-	-	$[inf,200,50,800]$
Y2	200	$[1,1,15]$	$i \in [n]$	$[50,Pois(500)]$	-	-	$[inf,200,50,800]$
Y3	200	$[1,1,15]$	$i \in [n]$	$[50,Unif(0:1000)]$	-	-	$[inf,200,50,800]$
V1	100	$[1,1,15]$	$i \in [n]$	$[100,700]$	-	-	$[inf,100,50,(400:3600)]$
V2	100	$[1,1,15]$	$i \in [n]$	$[100,500]$	-	-	$[inf,100,50,(400:3600)]$
V3	100	$[1,1,15]$	$i \in [n]$	$[50,700]$	-	-	$[inf,100,50,(400:3600)]$
V4	100	$[1,1,15]$	$i \in [n]$	$[100,700]$	-	-	$[inf,200,(10:500),800]$
V5	100	$[1,1,15]$	$i \in [n]$	$[100,500]$	-	-	$[inf,200,(10:500),800]$
V6	100	$[1,1,15]$	$i \in [n]$	$[50,700]$	-	-	$[inf,200,(10:500),800]$

† - Increase by specified amount in every cycle. "(+10)500" means increase by 10 until 500 is reached, "500(+10)" means increase by 10 starting with 500.

computation weights to be the multiples of a unit weight, and each platform's processing data and computation speeds to be the number of unit weights it may process in one second. Therefore the social costs are also measured in seconds.

A. Strategy Behaviour of Non-Cooperative Applications

In this group of experiments, we observe the behaviour of individual applications under different offload decision models.

1) Application with increasing weight: In this group of tests, we assume a system of 10 hybrid applications. We increase the weight of one of the applications (observed) while keeping all other (support) applications' weights unchanged. In S1 and S1F, as shown in Fig. 3 (a) and (b), we increase the computation weight of the observed application by 10 units until it reaches 500 at which point it has identical weights to the support applications. In S2, as shown in Fig. 3 (c) and (d), we begin with a group of 10 identical applications and gradually increase the computation weight of the observed application. The applications' non-cooperative offload strategies towards remote execution are as shown in Fig. 3 (a) and (c). The social costs are as shown in Fig. 3 (b) and (d).

Recall that when offload decisions are made according to incomplete information, all applications assume exclusive usage of the device's data connection. Because the wireless bandwidth in S1 and S1F are sufficiently large (50 units per second), the delay caused by this communication task is small enough to not deter the support applications (—⋆—) from remote execution. For the observed application (—○—), because its initial computation size is relatively small, unlike the applications in the support group, its benefit of remote execution is not sufficiently large enough to overcome the extra cost of data communication at early stages of S1 and S1F and prefers native execution.

On the contrary, when applications are given complete information of others' strategies, we see from Fig. 3 (a) that the observed application's preference on remote execution (—○—) is reduced as its computation weight increases.

This behaviour seems counterintuitive and counter-productive since it follows a completely opposite direction to that of the incomplete information scenario. Further reduction to (14) helps understand this strategy choice. We apply S1's application composition to (14) and derive

$$p_1^{\mathbb{R}} = \frac{8\left(\frac{w_1^d}{s_{\mathbb{R}}^d} + \frac{w_1^b}{s_{\mathbb{R}}^b}\right) + \frac{w_1^b}{s_{\mathbb{N}}^b} + 9\left(\frac{w_i^b}{s_{\mathbb{N}}^b} - \left(\frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b}\right)\right)}{9\left(\frac{w_1^d}{s_{\mathbb{R}}^d} + \frac{w_1^b}{s_{\mathbb{R}}^b} + \frac{w_1^b}{s_{\mathbb{N}}^b}\right)} \tag{21}$$

In (21) we see that $p_1^{\mathbb{R}}$ is dependent on both internal and external terms. When the platform parameters are fixed, the internal terms are influenced only by the weights of the observed application itself. The external term in (21) represents the *collective gain* that would have been obtained by other applications if they were to execute remotely.

In S1, S1F and S2, the external term is a constant since the weights of the support applications are constants. When w_1^b increases, the second internal term always increase faster than the first, the reduction in $p_1^{\mathbb{R}}$ (—○—) as shown in Fig. 3 (a) and (c) follows.

Note that at the first few test cycles, in S1, the external term dominates (21) and the observed application become a pure strategy agent with $p_1^{\mathbb{R}} = 1$.

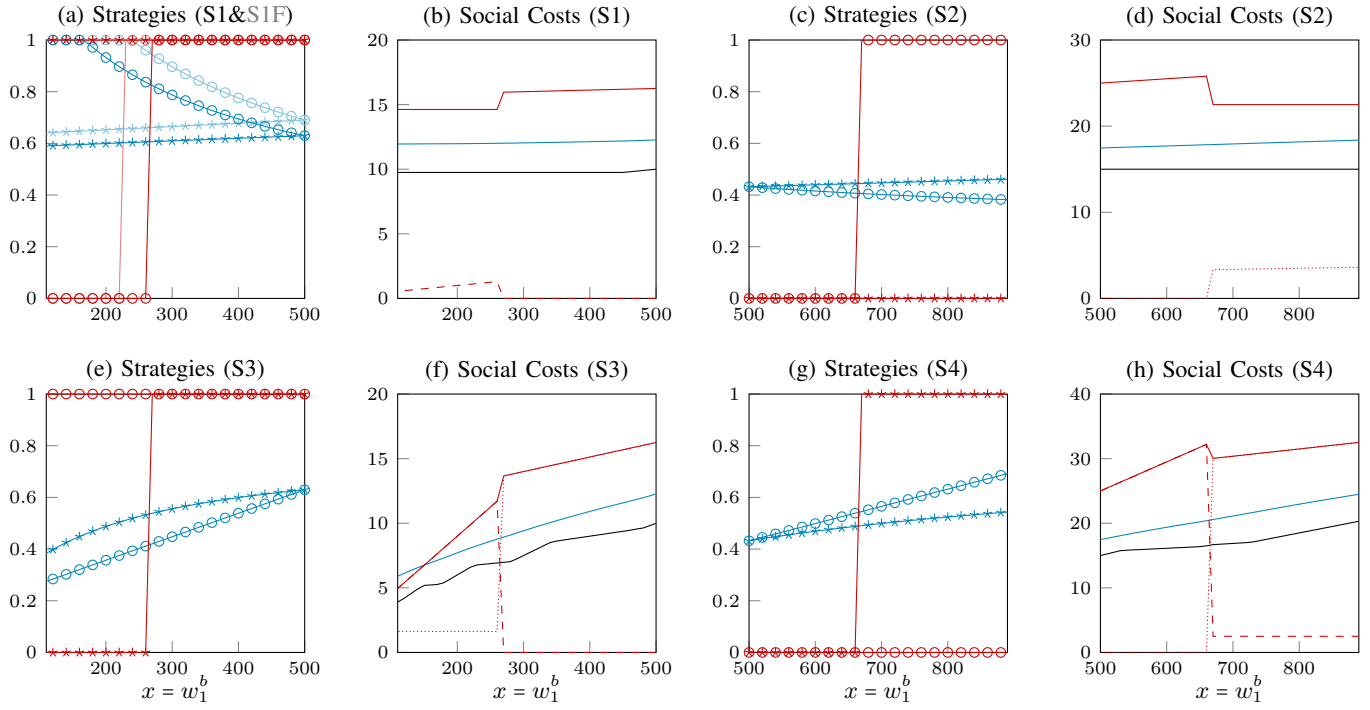

Fig. 3. Offload Strategy Behaviour. Non-Cooperative incomplete information: $a_1^{\mathbb{R}}$ —○—, $a_i^{\mathbb{R}}$ —⋆—, Θ_B ——,$\Theta_B^{\mathbb{N}}$ - - -,$\Theta_B^{\mathbb{R}}$ ·····; Non-Cooperative complete information (Nash): $p_1^{\mathbb{R}}$ —○—, $p_i^{\mathbb{R}}$ —⋆—, Θ_P —— and Cooperative Offload: Θ_{Opt} ——

2) Application within increasing weights: In S3 and S4, we fix the weight of the observed application and increase the support group's computation weight instead. In such cases, the external term in (21) become the variable. Because the increase in computation weight, the collective gain of the support group, i.e. the external term increases, and the increase in $p_1^{\mathbb{R}}$ (—○—) in Fig. 3 (e) and (g) follows.

Also note that because of the switch of behaviour between the observed application and the support group, their strategies (—○— and —⋆—) under incomplete information are swapped as shown in Fig. 3 (a) and (e) and Fig. 3 (c) and (g).

3) Change in Platform Parameters: In S1F, we double the computation speed of the remote platform, therefore the remote platform become more attractive to all applications as compared to S1. The results shown in Fig. 3 (a) matches our expectation. In the incomplete information scenario, the observed application (—○—) adopts remote execution earlier than in S1 (—○—). In the complete information offload game, all players shifted (from —○— and —⋆— to —○— and —⋆—) their strategy towards $\mathbb{R}$.

Also note that the external term dominated $p_1^{\mathbb{R}}$ for more number of cycles at the beginning of S1F than S1.

B. Social Costs of Decision Models

We now look at the social costs of different decision models of mobile cloud application ecosystems. As shown in Fig. 3 (b) (d) (f) and (h), in the incomplete information scenario, a step change is often observed because of the change of strategy by applications at certain thresholds. We plot the social cost (——) alongside the cost of $\mathbb{N}$ (- - -) and $\mathbb{R}$ (·····) to illustrate the relations between the makespan and the costs of each platform.

Compared with the other two decision models, the incomplete information model produces systems with highest social costs. Systems that are in Nash equilibrium as defined by the complete information game have higher social costs (——) than the optimal solution (——). We further observe that the gap (price of anarchy) between the optimal social costs and Nash social costs in Fig. 3 (d) and (h) increases while the gap between application computation weights increases. Therefore in the next group of tests, we investigate the relation between price of anarchy and the weight deviation in $[n]$.

C. Price of Anarchy

Recall that the price of anarchy of the complete information game is defined by the ratio between the Nash social cost (Θ_P) and the optimal social cost (Θ_{Opt}) of the system, which we denote with PoA_P. For comparison, we further define the price of anarchy in the symmetrically incomplete information game to be $\text{PoA}_B = \Theta_B : \Theta_{Opt}$. From S4 and S2, we observe slight increases in the price of anarchy when the difference in weight increases in $[n]$. This leads us to the hypothesis that the price of anarchy is more significant when the weights in $[n]$ have a high value of deviation.

1) Price of Anarchy and Application Weight Deviation in $[n]$: Following on the hypothesis, we conducted tests Y1, Y2 and Y3 (each occupies a column in Fig. 4). In these three groups of experiments, we run each cycle of our simulation with the same parameters with only the computation weights[6] of applications randomly drawn from three different

[6]We also conducted experiments that randomised both data and computation weights. The results are similar to that of Y1, Y2 and Y3 and so are omitted for brevity.

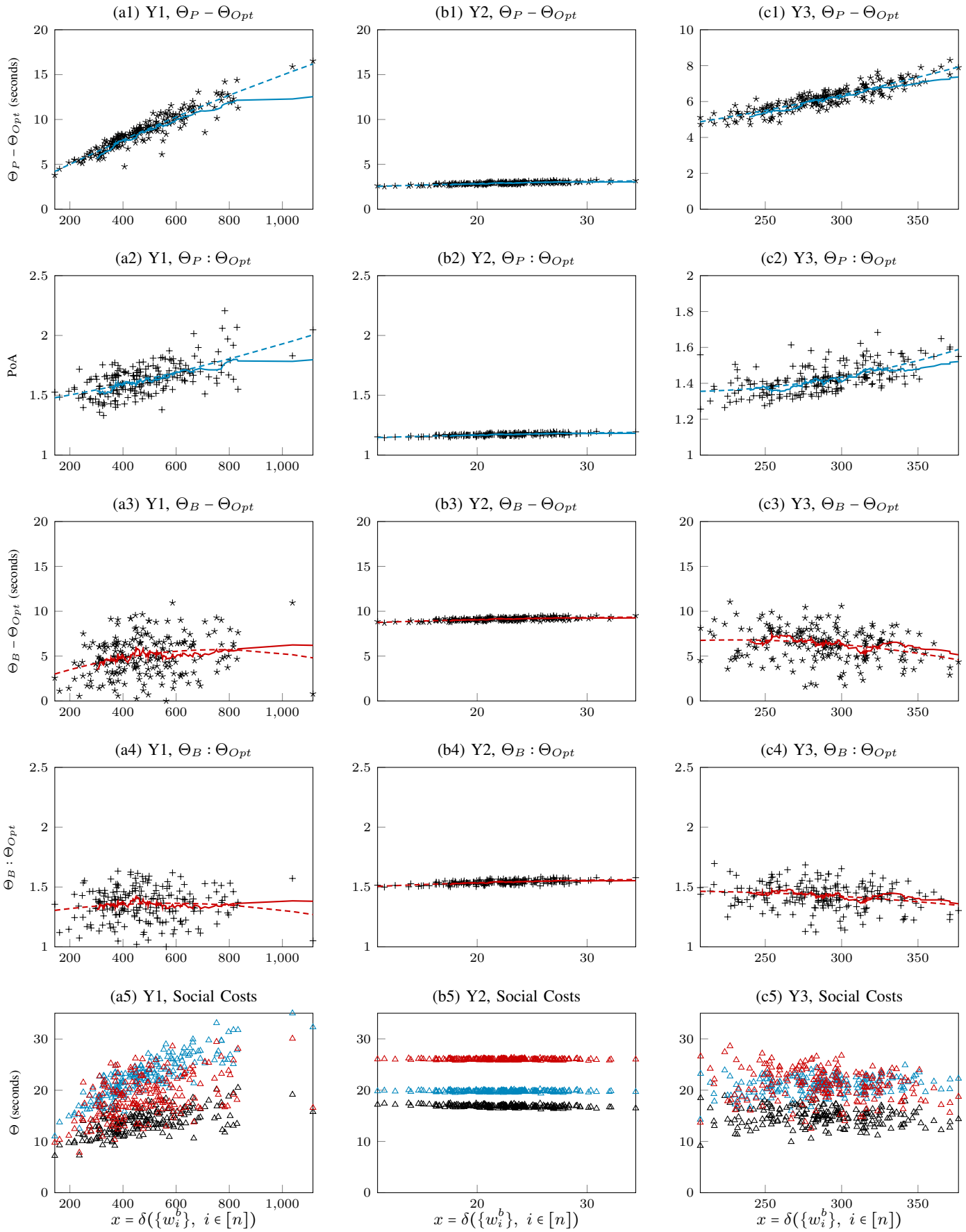

Fig. 4. Y1, Y2 and Y3. Gap between Θ_P, Θ_B and Θ_{Opt}, Price of Anarchy, Social Costs.

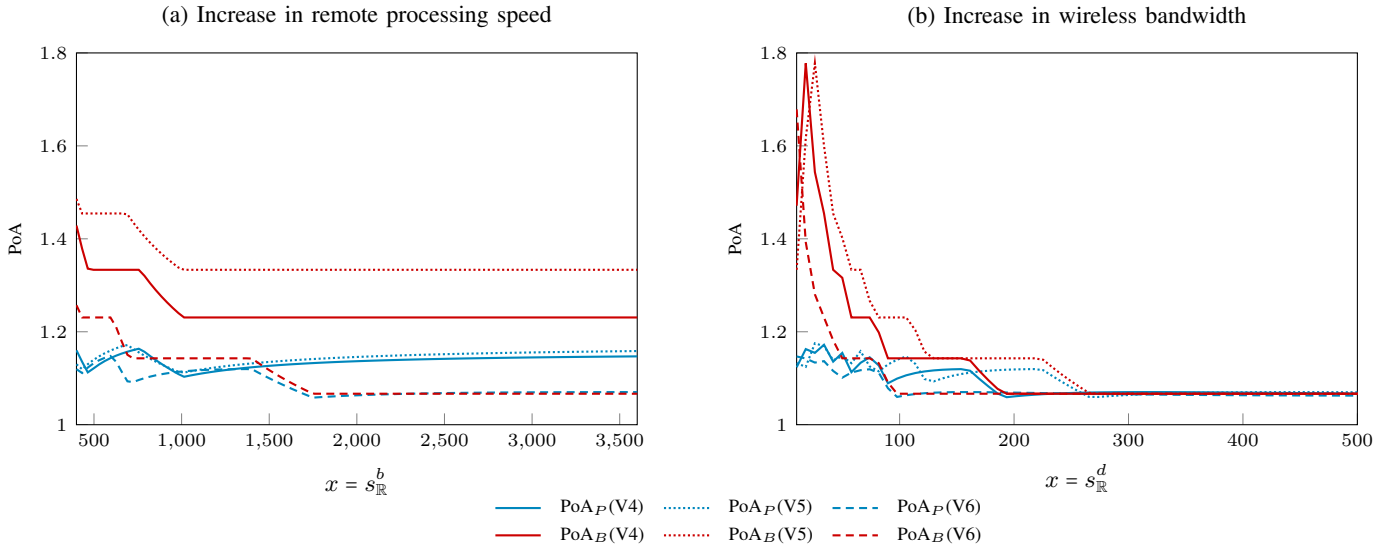

Fig. 5. Price of anarchy ($\Theta_P : \Theta_{Opt}$) following changes in platform parameters.

distributions (exponential, Poisson and uniform) at each test cycle. We choose these three distributions not only because of their difference in range and variance, but also because each distribution may be suitable to simulate the workload pattern of particular mobile application ecosystems. For instance, a set of applications whose workload depends on the arrival time of different user requests may be more suited to the exponential distribution. Applications whose workload is pre-defined to be within a range with equal probability to pick within this range is more suited to the uniform distribution model.

We label each test cycle with the standard deviation of the computation weights of all applications (i.e. $\delta(\{w_i^b\},\ i \in [n])$), and apply all three decision models to the system simulated in that cycle. With each of the two non-cooperative decision models, we record its social cost and compare it with the optimal social cost produced by the cooperative model. We plot five properties of the system against its deviation label in each of the five rows of Fig. 4. These properties includes Θ_P - Θ_{Opt}, price of anarchy = $\Theta_P : \Theta_{Opt}$, Θ_B - Θ_{Opt}, Θ_B:Θ_{Opt} and the social costs of the system in each test cycle.

From (a1) and (c1) of Fig. 4, we observe that the increase in application weight deviation (along the x-axis) indeed increase the probability of bigger gaps between Θ_P and Θ_{Opt}. The same trend is also observed in (a2) and (c2) for the price of anarchy albeit with a smaller gradient. In contrast, as shown in (b1) and (b2), all simulations in Y2 have a similar and stable price of anarchy. this is because Poisson distribution generates application weights with small deviations (c.f. range of x-axis in (b1)-(b5)). Furthermore, because applications simulated in Y2 are very similar to each other, the social costs of all three decision models are bounded within three small region as shown in (b5).

Row 3 and 4 of Fig. 4 illustrate the difference between Θ_B and Θ_{Opt}. While (b3) and (b4) follow a similar pattern as in (b1) and (b2), results from Y1 and Y3 are rather chaotic. This is due to the behaviour of the offload model based on incomplete information. Recall that the model predict an application's cost on both platforms based on incomplete information. This split is largely influenced by the device's bandwidth. When the bandwidth is given, this split is determined by the weights of the applications. When these weights are randomly chosen within a big range as in Y1, and Y3, this split of applications is likely to produce randomly unbalanced groups. Compared to the optimal split produced by the cooperative model, it is predictable that the Θ_B produced by this rather random behaviour has such random distance to Θ_{Opt}. We observe from (a3), (a4), (c3) and (c4) of Fig. 4 that as well as having a big distance from Θ_{Opt} (far from the x-axis), it is also possible for the incomplete information model to produce near optimal results (near to the x-axis).

The actual system costs of Y1-Y3 are shown in row 5 of Fig. 4. The increase in price of anarchy is most observable in (a5) for it has the greatest x range.

2) Price of Anarchy and Change in Platform Parameters: To further observe the price of anarchy in the system, we also conducted V1-V3 in which $s^b_{\mathbb{R}}$ is gradually increased in each test cycle, and V4-V6 in which $s^d_{\mathbb{R}}$ is gradually increased in each test cycle. As shown in Fig. 5, the price of anarchy in these tests are significantly lower than that from Y1 and Y3 because all applications have similar weights.

The increase in either processing speed and wireless bandwidth reduces and then stabilises the price of anarchy. This is because once a speed term is greater than a certain value, the cost term it is related to tends to zero and no longer have any effect over the system cost. Note that the turning points in Fig. 5 are caused when the optimal cooperative strategy switches one of the application's allocation from $\mathbb{N}$ to $\mathbb{R}$ as $\mathbb{R}$ becomes more and more attractive with its increasing computation speed (or wireless bandwidth).

V. Conclusion

The integration of mobile and cloud computing promises the user with convenient access to powerful applications at

any time and at any where. The research of computation offload and migration plays an essential part in this vision and ensures that this integration process is both seamless and efficient. In this paper, we investigate the efficiency of application offload in mobile cloud computing. We especially focus on the competition between mobile cloud applications residing on the same device which is overlooked by existing research under the topic.

Our main contribution is the game theoretic modelling of the non-cooperative offload game with complete information. This model is an extension to the classic load balancing game. We present detailed derivation of the mixed-strategy Nash equilibrium of this game. To compare the system's performance at equilibrium with existing computation offload mechanisms, we also model existing offload decision processes as a non-cooperative offload game with symmetrically incomplete information. Furthermore, we propose a cooperative scenario and solve the offload decision problem as a min-max integer program to obtain optimal offload schedules.

We compare the performance of all three offload decision models with a series of simulation experiments. On an application level, we observe the counterintuitive strategy decisions made by applications in the complete information game which help understand application behaviours when no global control is applied. On a system level, we discuss the price of anarchy in non-cooperative scenarios. We show that significant reduction in social cost can be obtained in a cooperative setting. The dependencies between price of anarchy and various system parameters are also investigated. We show that high deviation in application weights encourages high price of anarchy in non-cooperative scenarios.

Our study demonstrates the importance of recognising the potential competition between mobile cloud computing applications, and provide a suite of modelling tools to simulate and solve the offload decision problem in ecosystems of mobile cloud computing applications.

ACKNOWLEDGEMENT

This work is sponsored by the Research Project Grant of the Leverhulme Trust (Grant No. RPG-101) and the Key Program of National Natural Science Foundation of China Grant (No. 61133005, 61432005).

REFERENCES

[1] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making Smartphones Last Longer with Code Offload," in *MobiSys'10 The 8th International Conference on Mobile Systems, Applications, and Services*, Jun. 2010.

[2] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: elastic execution between mobile device and cloud," in *Proceedings of the sixth conference on Computer systems - EuroSys '11*, 2011, p. 301.

[3] S. Kosta, A. Aucinas, and R. Mortier, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *2012 Proceedings IEEE INFOCOM*, 2012, pp. 945–953.

[4] K. Kumar, "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.

[5] M. Zbierski and P. Makosiej, "Bring the Cloud to Your Mobile: Transparent Offloading of HTML5 Web Workers," in *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*. IEEE, Dec. 2014, pp. 198–203.

[6] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A Survey of Computation Offloading for Mobile Systems," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, 2012.

[7] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84–106, 2013.

[8] Google, "Our Mobile Planet," 2013.

[9] Nielsen, "Smartphones: So many apps, so much time," 2014.

[10] C. Shin, J.-H. Hong, and A. K. Dey, "Understanding and prediction of mobile application usage for smart phones," in *Proceedings of ACM Conference on Ubiquitous Computing, UbiComp'12*, 2012, pp. 173–182.

[11] E. Koutsoupias and C. Papadimitriou, "Worst-case equilibria," in *Proceedings of the 16th annual conference on Theoretical aspects of computer science, STACS'09*, 1999, pp. 404–413.

[12] U. Kremer, J. Hicks, and J. M. Rehg, "Compiler-directed remote task execution for power management," *Workshop on Compilers and Operating Systems for Low Power (COLP'00)*, 2000.

[13] Z. Li, C. Wang, and R. Xu, "Computation offloading to save energy on handheld devices," in *Proceedings of the international conference on Compilers, architecture, and synthesis for embedded systems - CASES '01*, 2001, p. 238.

[14] C. Wang and Z. Li, "Parametric analysis for adaptive computation offloading," in *Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation - PLDI '04*, vol. 39, no. 6, New York, New York, USA, Jun. 2004, p. 119.

[15] S. Kim, H. Rim, and H. Han, "Distributed execution for resource-constrained mobile consumer devices," *IEEE Transactions on Consumer Electronics*, vol. 55, no. 2, pp. 376–384, May 2009.

[16] J. Flinn and M. Satyanarayanan, "Balancing performance, energy, and quality in pervasive computing," in *Proceedings 22nd International Conference on Distributed Computing Systems*, 2002, pp. 217–226.

[17] R. K. Balan, M. Satyanarayanan, S. Y. Park, and T. Okoshi, "Tactics-based remote execution for mobile computing," in *Proceedings of the 1st international conference on Mobile systems, applications and services - MobiSys '03*, New York, New York, USA, 2003, pp. 273–286.

[18] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[19] M. A. Khan, "A survey of computation offloading strategies for performance improvement of applications running on mobile devices," *Journal of Network and Computer Applications*, vol. 56, pp. 28–40, 2015.

[20] B. Gao, L. He, L. Liu, K. Li, and S. Jarvis, "From Mobiles to Clouds: Developing Energy-Aware Offloading Strategies for Workflows," in *Proceedings of the 13th ACM/IEEE International Conference on Grid Computing (GRID'12)*, 2012, pp. 139 –146.

[21] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo : a Computation Offloading Framework for Smartphones," in *MOBICASE 2010 IEEE Computer Society*, 2010.

[22] C. Chekuri and M. Bender, "An Efficient Approximation Algorithm for Minimizing Makespan on Uniformly Related Machines," in *Proceedings of the 6th Conference on Integer Programming and Combinatorial Oprimization*, 1998.

[23] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference - IMC '09*, 2009, p. 280.

[24] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Optimal cache allocation for Content-Centric Networking," in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, Oct. 2013, pp. 1–10.

APPENDIX A
DERIVATION OF $p_i^{\mathbb{R}}$

When a game is in a state of mixed-strategy equilibrium, we have $\mathbb{E}[c_i^{\mathbb{R}}] = \mathbb{E}[c_i^{\mathbb{N}}]$. This with (6) we get

$$\mathbb{E}[C_{\mathbb{R}}] + (1-p_i^{\mathbb{R}})\left(\frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b}\right) = \mathbb{E}[C_{\mathbb{N}}] + (1-p_i^{\mathbb{N}})\left(\frac{w_i^d}{s_{\mathbb{N}}^d} + \frac{w_i^b}{s_{\mathbb{N}}^b}\right) = \mathbb{E}[C_{\mathbb{N}}] + p_i^{\mathbb{R}}\left(\frac{w_i^d}{s_{\mathbb{N}}^d} + \frac{w_i^b}{s_{\mathbb{N}}^b}\right)$$

$$p_i^{\mathbb{R}}\left(\frac{w_i^d}{s_{\mathbb{N}}^d} + \frac{w_i^b}{s_{\mathbb{N}}^b} + \frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b}\right) - \left(\frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b}\right) = \mathbb{E}[C_{\mathbb{R}}] - \mathbb{E}[C_{\mathbb{N}}] \tag{22}$$

For applications that are fixed to run on either $\mathbb{N}$ or $\mathbb{R}$, i.e. $i \in [n]^{\mathbb{N}} \cup [n]^{\mathbb{R}}$ we define

$$C_j^f = \sum_{i\in[n]^j}\left(\frac{w_i^d}{s_j^d} + \frac{w_i^b}{s_j^b}\right), \quad j \in \{\mathbb{N}, \mathbb{R}\} \tag{23}$$

Take this into (4) we have

$$\mathbb{E}[C_{\mathbb{N}}] = C_{\mathbb{N}}^f + \sum_{i\in[n]-[n]^{\mathbb{N}}} p_i^{\mathbb{N}}\left(\frac{w_i^d}{s_{\mathbb{N}}^d} + \frac{w_i^b}{s_{\mathbb{N}}^b}\right) \quad \text{and} \quad \mathbb{E}[C_{\mathbb{R}}] = C_{\mathbb{R}}^f + \sum_{i\in[n]-[n]^{\mathbb{R}}} p_i^{\mathbb{R}}\left(\frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b}\right) \tag{24}$$

Take these into (11) we have

$$\mathbb{E}[C_{\mathbb{N}}] = C_{\mathbb{N}}^f + \sum_{i\in[n]-[n]^{\mathbb{N}}} a_i^{\mathbb{N}}\left(\mathbb{E}[C_{\mathbb{N}}] - \mathbb{E}[c_i^{\mathbb{N}}] + \left(\frac{w_i^d}{s_{\mathbb{N}}^d} + \frac{w_i^b}{s_{\mathbb{N}}^b}\right)\right) \tag{25}$$

$$\mathbb{E}[C_{\mathbb{R}}] = C_{\mathbb{R}}^f + \sum_{i\in[n]-[n]^{\mathbb{R}}} a_i^{\mathbb{R}}\left(\mathbb{E}[C_{\mathbb{R}}] - \mathbb{E}[c_i^{\mathbb{R}}] + \left(\frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b}\right)\right) \tag{26}$$

Take a difference between these two equations we have

$$\mathbb{E}[C_{\mathbb{R}}] - \mathbb{E}[C_{\mathbb{N}}] = C_{\mathbb{R}}^f - C_{\mathbb{N}}^f + |[n]^{\mathbb{H}}|(\mathbb{E}[C_{\mathbb{R}}] - \mathbb{E}[C_{\mathbb{N}}]) + \sum_{k\in[n]^{\mathbb{H}}}\left(\frac{w_k^d}{s_{\mathbb{R}}^d} + \frac{w_k^b}{s_{\mathbb{R}}^b}\right) - \left(\frac{w_k^d}{s_{\mathbb{N}}^d} + \frac{w_k^b}{s_{\mathbb{N}}^b}\right) \tag{27}$$

$$\mathbb{E}[C_{\mathbb{R}}] - \mathbb{E}[C_{\mathbb{N}}] = \left(C_{\mathbb{R}}^f - C_{\mathbb{N}}^f + \sum_{k\in[n]^{\mathbb{H}}}\left(\frac{w_k^d}{s_{\mathbb{R}}^d} + \frac{w_k^b}{s_{\mathbb{R}}^b}\right) - \left(\frac{w_k^d}{s_{\mathbb{N}}^d} + \frac{w_k^b}{s_{\mathbb{N}}^b}\right)\right)\Big/\left(1 - |[n]^{\mathbb{H}}|\right) \tag{28}$$

Finally, compare this with (22) we get

$$p_i^{\mathbb{R}}\left(\frac{w_k^d}{s_{\mathbb{N}}^d} + \frac{w_k^b}{s_{\mathbb{N}}^b} + \frac{w_k^d}{s_{\mathbb{R}}^d} + \frac{w_k^b}{s_{\mathbb{R}}^b}\right) - \left(\frac{w_k^d}{s_{\mathbb{R}}^d} + \frac{w_k^b}{s_{\mathbb{R}}^b}\right) = \left(C_{\mathbb{R}}^f - C_{\mathbb{N}}^f + \sum_{k\in[n]^{\mathbb{H}}}\left(\frac{w_k^d}{s_{\mathbb{R}}^d} + \frac{w_k^b}{s_{\mathbb{R}}^b}\right) - \left(\frac{w_k^d}{s_{\mathbb{N}}^d} + \frac{w_k^b}{s_{\mathbb{N}}^b}\right)\right)\Big/\left(1 - |[n]^{\mathbb{H}}|\right) \tag{29}$$

$$\begin{aligned} p_i^{\mathbb{R}} = {} & \left(\frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b}\right)\Big/\left(\frac{w_i^d}{s_{\mathbb{N}}^d} + \frac{w_i^b}{s_{\mathbb{N}}^b} + \frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b}\right) \\ & + \left(C_{\mathbb{R}}^f - C_{\mathbb{N}}^f + \sum_{k\in[n]^{\mathbb{H}}}\left(\frac{w_k^d}{s_{\mathbb{R}}^d} + \frac{w_k^b}{s_{\mathbb{R}}^b}\right) - \left(\frac{w_k^d}{s_{\mathbb{N}}^d} + \frac{w_k^b}{s_{\mathbb{N}}^b}\right)\right)\Big/\left(\left(1 - |[n]^{\mathbb{H}}|\right)\left(\frac{w_i^d}{s_{\mathbb{N}}^d} + \frac{w_i^b}{s_{\mathbb{N}}^b} + \frac{w_i^d}{s_{\mathbb{R}}^d} + \frac{w_i^b}{s_{\mathbb{R}}^b}\right)\right) \end{aligned} \tag{30}$$